The Development of a Linus-Compatible Slow Control Interface for the MINERvA Data Acquisition Electronics

Christopher Marshall

Office of Science, Science Undergraduate Laboratory Internship (SULI) Program

Carleton College

Northfield, Minnesota

Fermi National Accelerator Laboratory

Batavia, Illinois

July 28, 2009

Prepared in partial fulfillment of the requirement of the Office of Science, Department of Energy's Science Undergraduate Laboratory Internship under the direction of David Boehnlein in the Particle Physics Division at Fermi National Accelerator Laboratory.

Participant: ______________________________

Signature

Research Advisor: ______________________________

Signature

ABSTRACT

The Development of a Linus-Compatible Slow Control Interface for the MINERvA Data Acquisition Electronics

Christopher Marshall (Carleton College, Northfield, MN 55057), David Boehnlein (Fermi National Accelerator Laboratory, Batavia, IL 60510).

As the MINERvA neutrino-nucleus interaction experiment at Fermilab transitions from its tracking prototype detector to the full detector, its data acquisition system shifts from Microsoft Windows to Scientific Linux. This change provides performance benefits for the experiment's detector readout and alleviates some cross-platform compatibility issues. However, the change in operating system brings about a need for a new readout library to access information in the detector's electronics, and an accompanying graphical user interface. This interface, the slow control, allows users to send messages to the MINERvA detector and configure its electronics for a run. A Linux-compatible slow control interface was developed at Fermilab in June and July 2009. Written in Python, it uses the wxPython library and communicates with the readout library, enabling users to monitor the data acquisition electronics in an environment free of programming syntax.

Introduction

The MINERvA experiment at the Fermi National Accelerator Laboratory will employ a fine-grained, solid scintillation detector to make a variety of precision neutrino-nucleus interaction measurements. Unlike the MINOS near detector, which sits immediately downstream in the Neutrinos at Main Injector (NuMI) Hall, MINERvA will feature a fully-active inner detector, made up of 30,272 triangular plastic scintillator strips [1]. Each of these strips reads out on one side to a multi-anode photomultiplier tube (PMT). Neutrino events in the detector create particles which scintillate in the plastic, and the signal propagates down a wavelength-shifting fiber-optic cable until it reaches the electronics. The 473 PMTs amplify the signal and digitize the optical light inside a light-tight box, outside of which sits the controlling front-end board (FEB) [1].

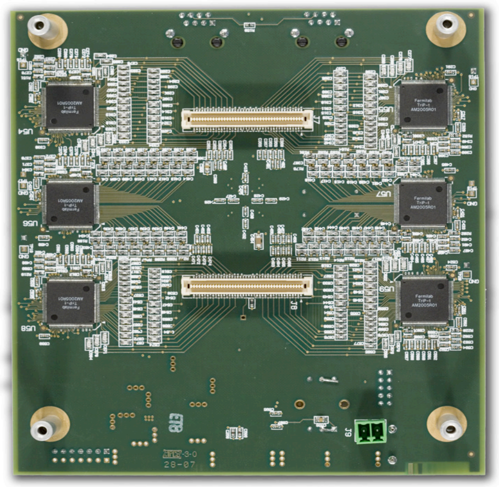


Figure 1: The rear view of the MINERvA front-end board shows the six TriP chips. The FPGA sits on the reverse side in the center.

Each board contains a central field-programmable gate array (FPGA), which manages six trigger pipeline (TriP) chips, three on each side of the board. The TriPs, originally designed for the fiber tracker of the DØ experiment at Fermilab, control the gain and threshold voltages of the PMT's 64 channels [2]. In addition to controlling the TriPs, the FPGA also directs the high-voltage settings. Up to ten FEBs can be daisy-chained together to form a chain readout controller (CROC) module. The CROCs and their parent CROC Interface Modules (CRIMs) sit inside one of MINERvA's two VME crates and are operated by a commercial CAEN card.

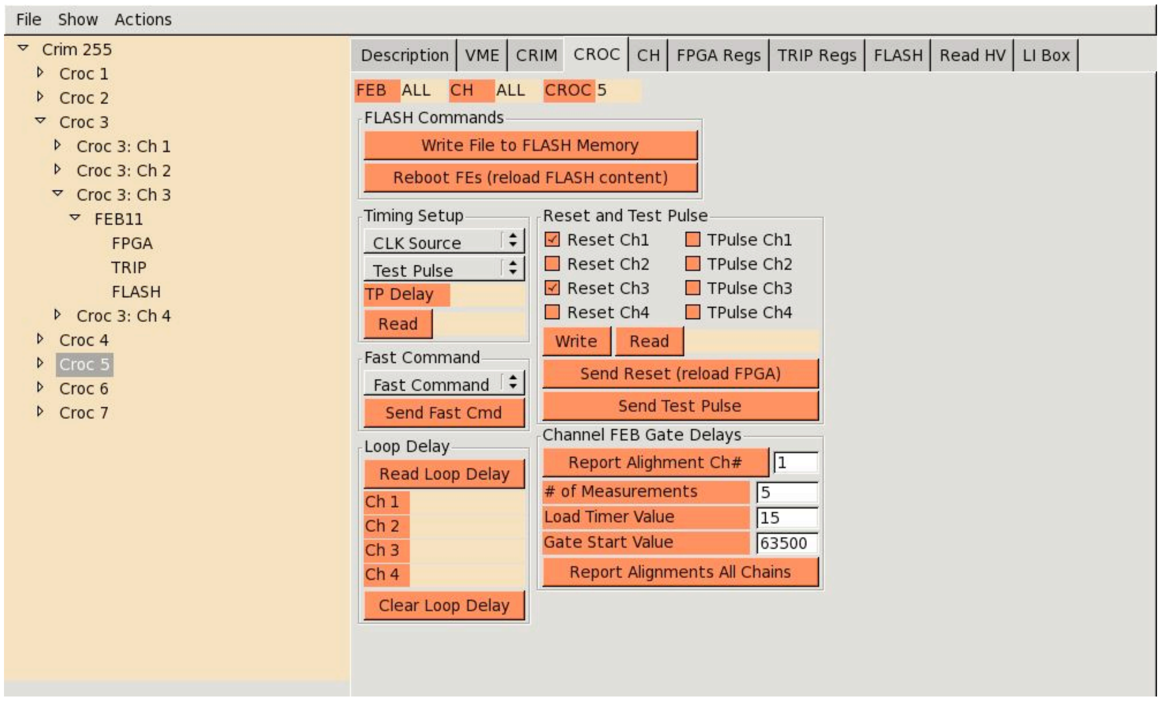
The VME crates connect to the two-CPU data acquisition (DAQ) computer, located underground near the detector. While the current DAQ uses some custom software developed specifically for MINERvA, it also takes advantage of CAEN commercial programs and software developed for the Large Hadron Collider beauty (LHCb) experiment [3]. One CPU of the DAQ underground computer receives real-time data from the detector; the other communicates with the MINERvA slow control system by receiving “frames.” Essentially messages to the detector, these frames allow users in the control room at Fermilab’s Wilson Hall to configure the electronics for a run. The slow control manages all components of the electronics system; some settings pertain to entire CROCs while others affect only specific FPGA or TriP chips. A readout library provides access to the slow control settings, allowing users to read information from the system, alter the settings if desired, and write the new settings to memory in the electronics. The process of reading and writing configuration settings occurs through a graphical user interface (GUI) in the control room computers.

The tracking prototype (TP) consisted of about one-fourth of the full MINERvA detector. After construction and testing on the surface, the prototype’s first modules moved into the NuMi hall in March 2009, and the TP phase of the experiment ran until the Fermilab beam shutdown in June 2009. Beyond testing and debugging the detector itself, the TP run served as a test for the DAQ, which read out data from the TP just as it will the full detector [4]. During this phase, the DAQ ran on Microsoft Windows. For the full detector, MINERvA will use a new Linux DAQ for three reasons. First, nearly all MINERvA computing takes place on a Scientific Linux platform. Second, LHCb may discontinue support for Windows, which would leave MINERvA using old, unsupported software. Third, tests indicate that a Linux DAQ could run an order of magnitude faster than its Windows counterpart [5].

In addition to writing a new DAQ for Linux, the platform shift brings about a need for a new slow control readout library and GUI. For the Windows DAQ, both the readout library and the GUI were written in C# [6]. These programs run on Windows only and have no cross-platform portability. The core of a C++ readout library was developed in anticipation of a shift to Linux-based DAQ computing, written primarily by Elaine Schulte of Rutgers University. The C# slow control GUI used during the TP run could work with the C++ library, but will not run on Linux computers.

METHODS

Python was chosen as the programming language for the Linux DAQ slow control GUI for several reasons. Most importantly, the availability of well-documented GUI graphics libraries for Linux made design simple. The simplicity of Python code makes the source both easy to read and easy to modify in the event of a future desire for enhanced functionality. Python ports easily across platforms and has multiple open-source packages that generate wrapper code for C++ functions. Performance, while a concern for the slow control, was not an issue for the GUI, which performs no real-time data acquisition. Also, the demands on the GUI itself are relatively small; the largest task involves saving or loading all of the settings, which in practice will not exceed a few thousand fields.



FEB ALL CH ALL CROC 5

☑ Reset Ch1
☐ Reset Ch2
☑ Reset Ch3
☐ Reset Ch4

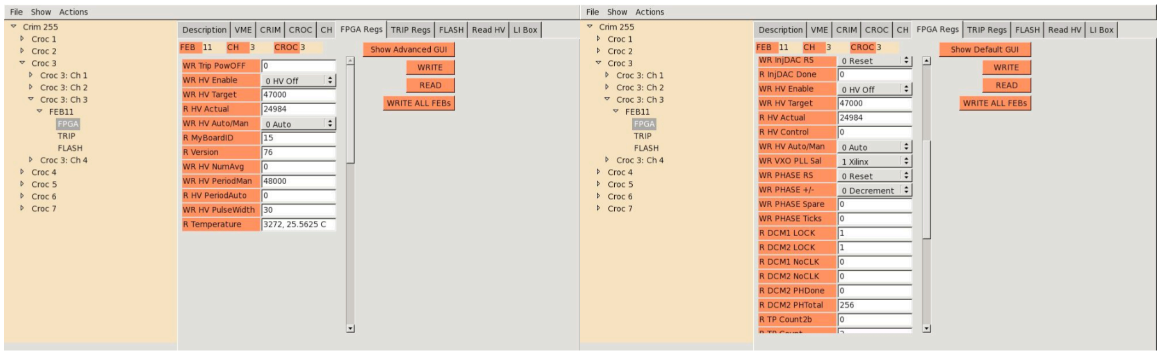
Report Alignment Ch#	1
# of Measurements	5
Load Timer Value	15
Gate Start Value	63500

Figure 2: The CROC settings of the MINERvA DAQ slow control GUI. The tree control at left accesses the GUI page at right.

Python's default GUI library, Tk, lacks several features necessary for the slow control GUI. The C# Slow Control GUI, developed by Fermilab's Cristian Gingu, organized the electronics into a tree structure. The logic of this interface mimics the hierarchy of the actual electronics; the parent-child relationship in the tree mirrors the flow of information in the detector. The tree control does not exist in Tk. Also, the slow control GUI manages too much information to fit on a single page, and Tk does not have a built-in tab method. Both of these methods exist in another GUI library, wx [7]. Though not as well documented or as widely used as Tk, the wx library was chosen for its functionality.

RESULTS

The wx library has many dependencies, mostly graphics libraries. MINERvA's computers lacked ten packages, which were built from source code for Scientific Linux in June 2009. With the wx library in place, the development of the slow control GUI began in June and the shell was completed on July 21, 2009. The code (before plugging in functions from the readout library) consists of 1 842 lines, divided into 19 Python classes. Each tab in the GUI notebook manages one component of the slow control. A user navigates the interface by selecting pages from the tree. Clicks to the tree instruct the GUI to display information for the desired electronic component. The tree control allows the user to access all hardware (CRIMs, CROCs, FPGAs and TriPs) and save settings for every element separately, all in a compact format.

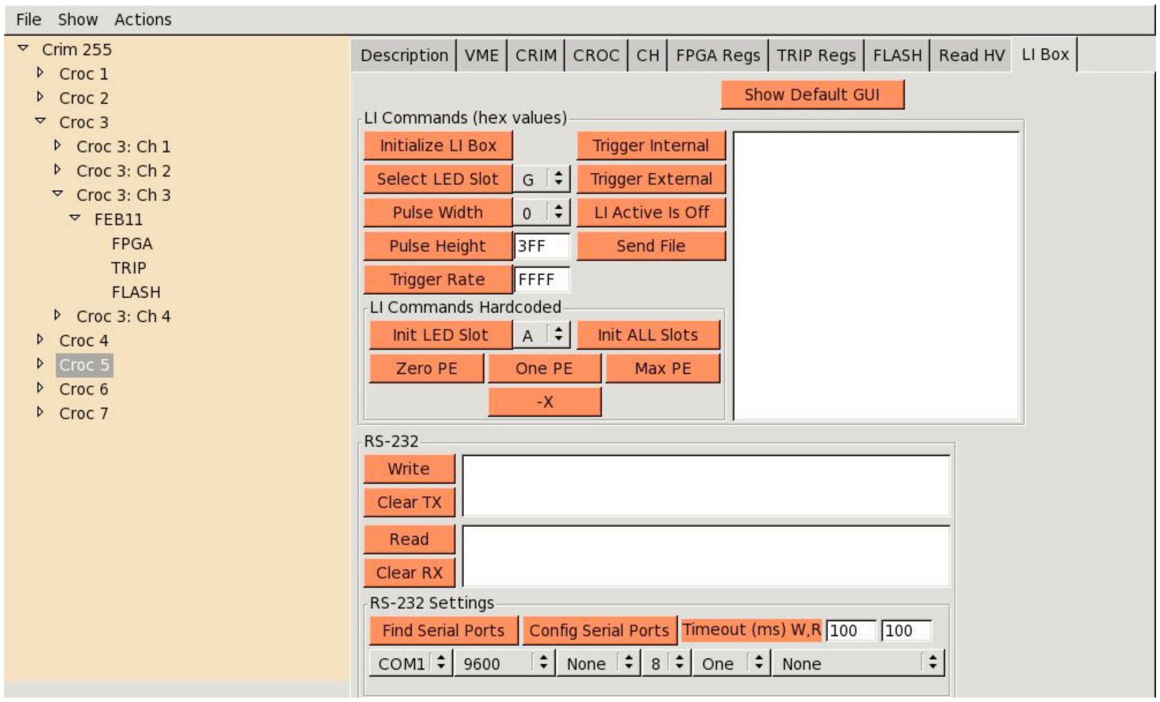


FEB	11
CH	3
CROC	3
WR Trip PowOFF	0
WR HV Enable	0 HV Off
WR HV Target	47000
R HV Actual	24984
WR HV Auto/Man	0 Auto
R MyBoardID	15
R Version	76
WR HV NumAvg	0
WR HV PeriodMan	48000
R HV PeriodAuto	0
WR HV PulseWidth	30
R Temperature	3272, 25.5625 C

FEB	11
CH	3
CROC	3
WR InjDAC RS	0 Reset
R InjDAC Done	0
WR HV Enable	0 HV Off
WR HV Target	47000
R HV Actual	24984
R HV Control	0
WR HV Auto/Man	0 Auto
WR VXO PLL Sal	1 Xilinx
WR PHASE RS	0 Reset
WR PHASE +/-	0 Decrement
WR PHASE Spare	0
WR PHASE Ticks	0
R DCM1 LOCK	1
R DCM2 LOCK	1
R DCM1 NoCLK	0
R DCM2 NoCLK	0
R DCM2 PHDone	0
R DCM2 PHTotal	256
R TP Count2b	0

Figure 3: The default GUI for the FPGA Register (left) hides many of the options.

For the FPGA and TriP settings, the user can toggle back and forth between a default and an advanced mode; in default mode the GUI hides the fields that will see less use while storing the data they contain. The GUI can also be used to read and monitor high-voltage settings and to control MINERvA's light injection (LI) system. The LI box contains 20 light-emitting diodes (LEDs), which inject light into individual PMTs for testing and calibration. Users can initialize and configure LI box runs with the slow control GUI. Because of the amount of data the slow control GUI contains, users may want to save all settings for later use. The interface can save and load all of its fields to human-readable and human-editable text files through an option in the File menu. This output format also enables search-and-replace editing, which is impossible in the framework of the GUI.



Select LED Slot: G
Pulse Width: 0
Pulse Height: 3FF
Trigger Rate: FFFF
Init LED Slot: A
Timeout (ms) W,R: 100 100
COM1 | 9600 | None | 8 | One | None

Figure 4: The advanced mode of the LI box tab. MINERvA's light injection system can be run from the slow control.

FUTURE WORK

In order to successfully implement the slow control into MINERvA's electronics and DAQ system, the readout library must be completed. Then, the readout functions can be hooked into the GUI. The first step involves continued expansion and debugging of the existing C++ readout library. The second step requires wrapper functions to convert the output from the C++ functions into Python objects. Then, the C++ classes can be imported into the Python GUI code directly, granting the interface access to variables extracted from the detector by the readout functions. Manually generating this code would entail writing a wrapper function for each C++ class. Instead, SWIG, an open-source C/C++ wrapper code generator, can produce the necessary functions with minimal work by hand.

ACKNOWLEDGEMENTS

This work would not have been possible without the help of Gabe Perdue. The C# GUI written by Cristian Gingu provided plenty of inspiration. Also, thanks to Dave Boehnlein for his support, and to the MINERvA Collaboration. Funding for this work was provided by the United States Department of Energy, through the Science Undergraduate Laboratory Internship (SULI) program.

REFERENCES

[1] MINERvA Collaboration, "The MINERvA Technical Design Report," December 2006.

[2] MINERvA Collaboration, "Impact of MINERvA Design, Assembly and Installation on Femilab," March 2004.

[3] G. N. Perdue, "DAQ Status," MINERvA-doc-3749, June 2009.

[4] J. Castorena, "Commissioning of the MINERvA Tracking Prototype," MINERvA-doc-3653, May 2009.

[5] G. N. Perdue, "Plans for the Once and Future MINERvA DAQ," MINERvA-doc-3537, April 2001.

[6] C. Gingu and G. N. Perdue, "Slow Control Manual," MINERvA-doc-3238, January 2009.

[7] R. Dunn, "What is wxPython?" July 2009, http://www.wxpython.org/what.php.